MovieLens Rating Prediction Project

Edward Amankwah

2019-11-07

Contents

Introduction

Machine learning is a branch of artificial intelligence where statistical methods are used to help a system to improve at tasks with training and experience. One application of machine learning includes the review of previous spending habits to give personalised recommendations or the ability to make recommendations of items to general users and customers.

Companies such as Amazon, eBay, Alibaba sell diverse products to many customers and allow them to rate their products. These companies end up collecting massive datasets on user recommendations on a particular item. Items with high predicted ratings are then recommended to potential users. The success of Nexflix has been attributed to its strong movie recommendation system.

This report is subdivided into introduction which includes motivation of the project and data preprocessing, methods and analysis, modelling approach, results, discussion, conclusion and references.

Motivation of Project

This project is part of the final capstone project to complete the HarvardX professional data science certification program. The project aims at developing a machine learning algorithm capable of predicting movie ratings by a user based on users past rating of movies (movies are rated from 0.5 star to 5 stars).

Evaluation Metric

The evaluation metric for the algorithm performance is the Root Mean Square Error (RMSE). RMSE is one of the most used measure of the differences between values predicted by a model and the values observed. RMSE is a measure of accuracy,which compares the differences between values predicted by a model and the values observed for a particular dataset. The lower the RMSE the better the prediction accuracy. RMSE is sensitive to outlier since it is proportional to the size of the squared error. The RMSE will be used to assess the quality of models that will be developed. The best resulting model with RMSE less than or equal to 0.8649 will be used to predict the movie ratings.

Dataset

The MovieLens dataset used for this project can be downloaded from the links below:

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

Data Preprocessing

The larger MovieLens dataset from Netflix data is not publicly available. However, the The GroupLens research lab generated their own database which has over 20 million ratings for over 27,000 movies by more than 138,000 users. A smaller subset of about 10 million ratings was used is this project.

```
# Note: this process could take a couple of minutes

############################################################
# Create edx set, validation set, and submission file
############################################################
# Note: this process could take a couple of minutes for loading required package: tidyverse and package
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
col.names = c("userId", "movieId", "rating", "timestamp"))
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
title = as.character(title),
genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

Train and Validation Sets

The code below was be used to generate the datasets. The algorithm was developed using the train set(edx subset only) and the final test of the algorithm predicted movie ratings using the validation subset assuming they were unknown.

```
# The Validation subset will be 10% of the MovieLens data.
set.seed(1, sample.kind="Rounding")
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

#Make sure userId and movieId in validation set are also in edx subset:
validation <- temp %>%
semi_join(edx, by = "movieId") %>%
semi_join(edx, by = "userId")
# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

MovieLens Data Summary

The first six rows of "edx" subset below depict six variables "userID", "movieID", "rating", "timestamp", "title", and "genres". Each of the rows indicate a single rating of a user for a single movie.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

A summary of the edx subset shows that there are no missing values.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

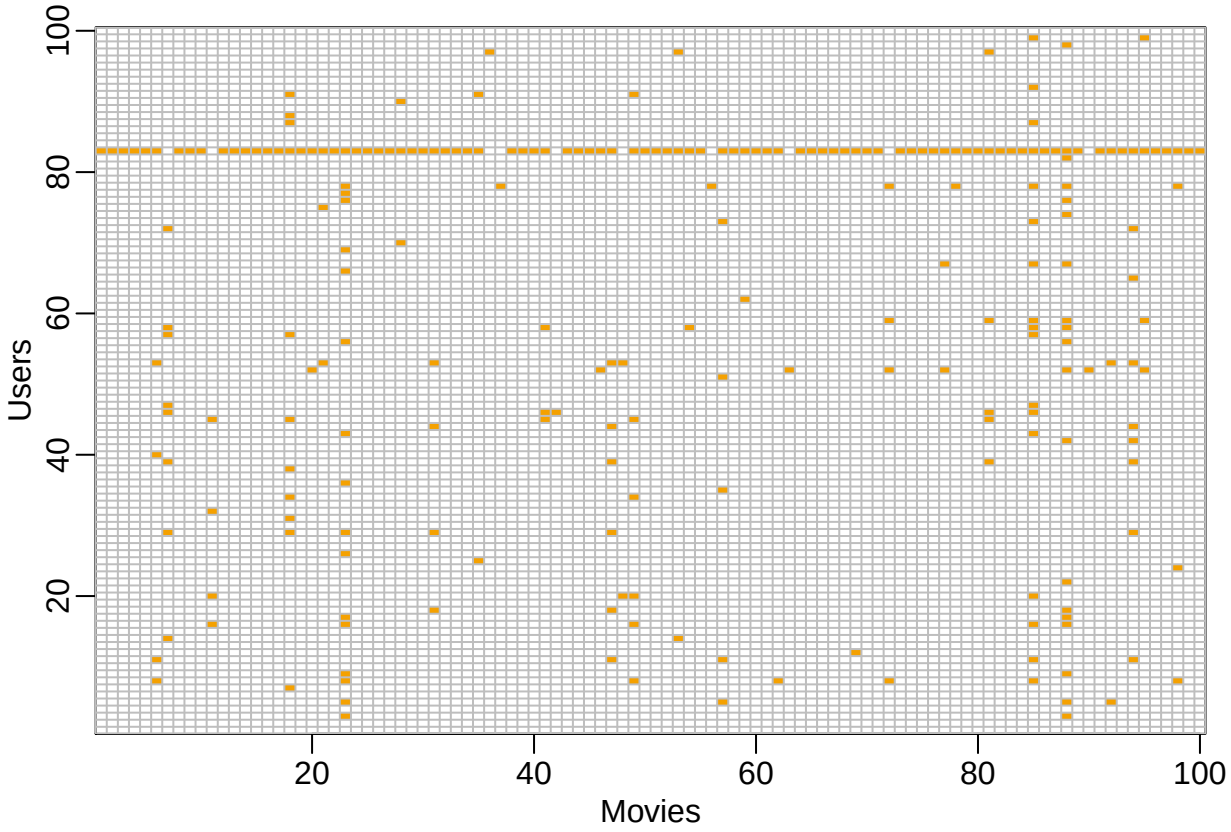
MovieLens Data Exploration

The number of distinct movies and users in the training set, edx is shown below:

```
##   n_users n_movies
## 1   69878    10677
```

The product of the two numbers (746,087,406) is far greater than 10M, yet the data set under consideration has about 10 million rows indicating that not all users may have rated all movies.

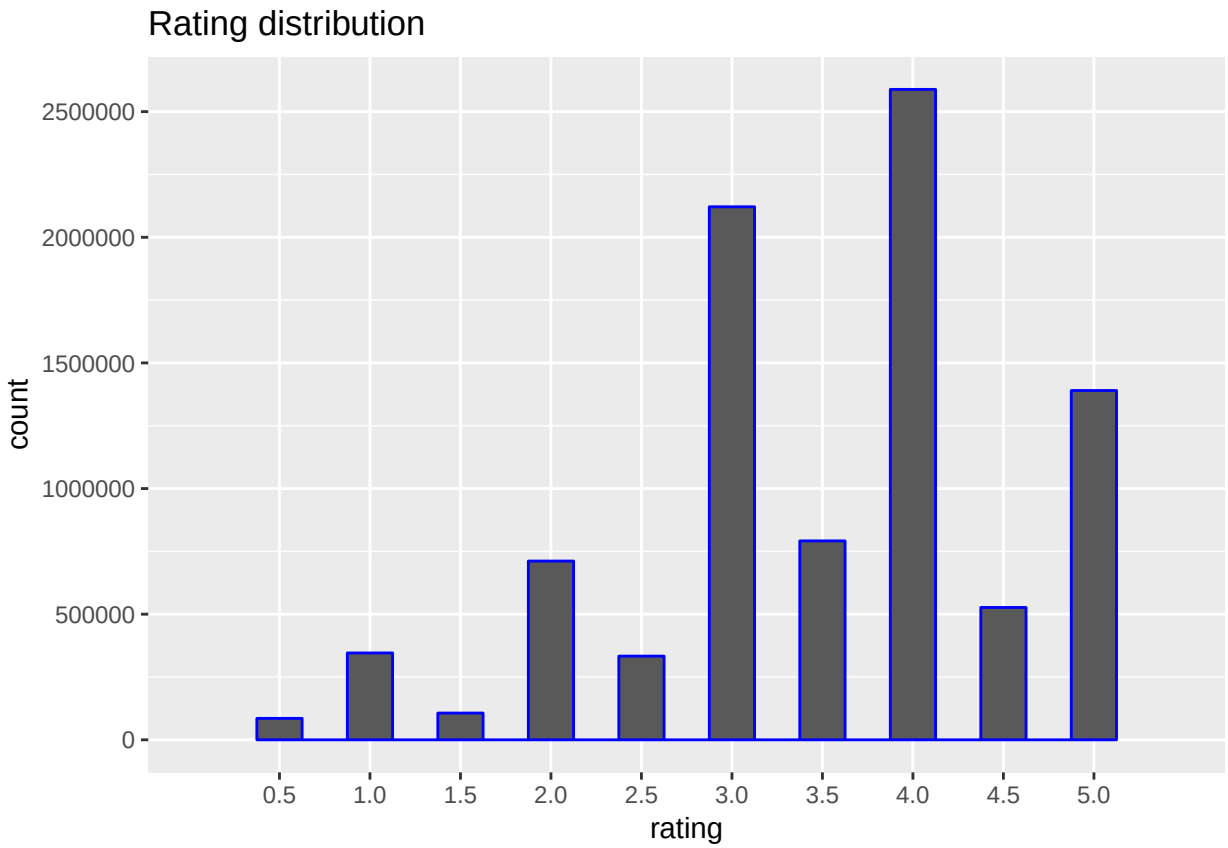
These datasets can be thought of as a very large matrix, with users on the rows and movies on the columns, with many empty cells. The task of a recommendation system can then be considered as filling the empty cells. To visualize how sparse the matrix is, matrix for a random sample of 100 movies and 100 users with yellow indicating a user/movie combination for which there is a rating is shown below.



Note that if the rating is predicted for movie m by user u, in principle, all other ratings related to movie m and by user u may be used as predictors, but different users rate different movies and a different number of movies. In essence, the entire matrix can be used as predictors for each cell.

Methods and Analysis

Ratings Distributions

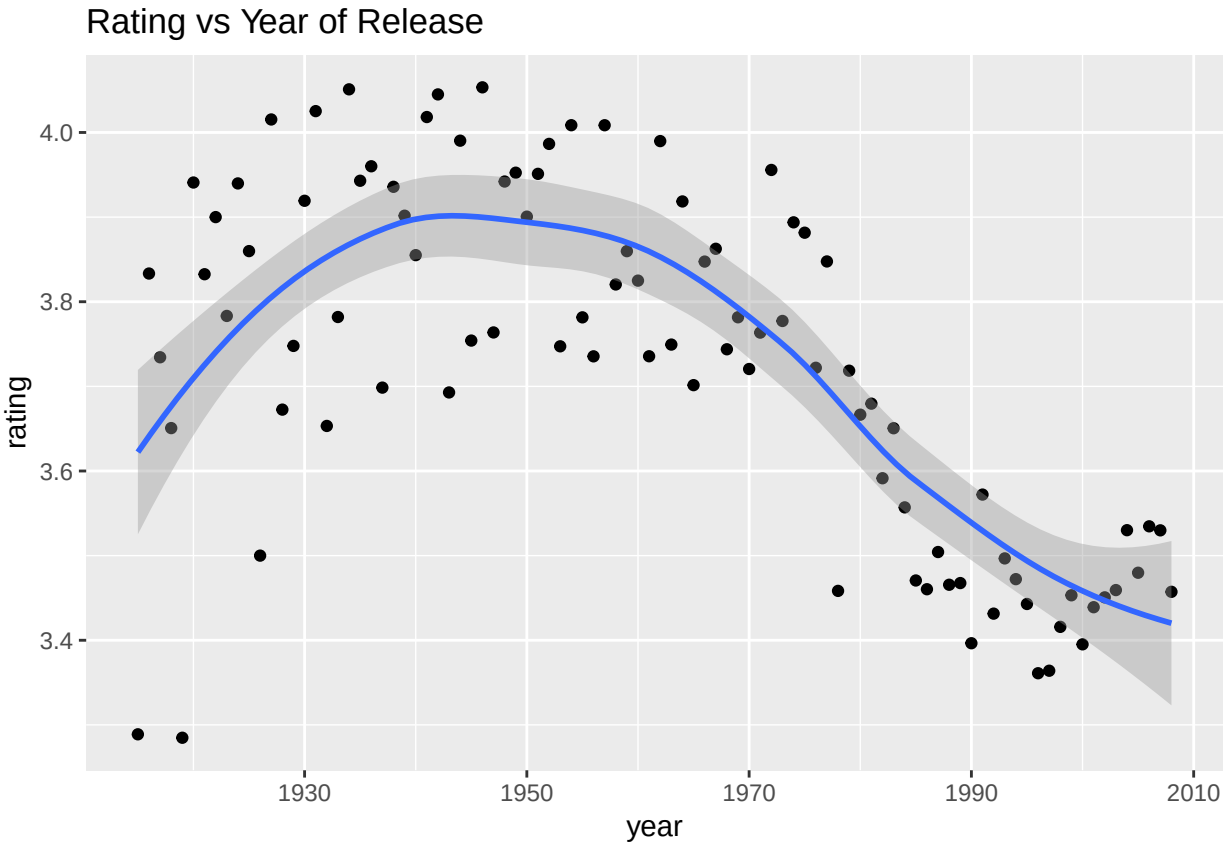


The above rating distribution indicates that users have rated some movies often higher than others while others have few ratings. This may mean low rating numbers might lead to inaccurate estimate for predictions. This means further exploration of the effect of different features would be needed to make good predictive models

Ratings vs Year of Release

The general trend of movie viewers and their rating habits can be shown below:

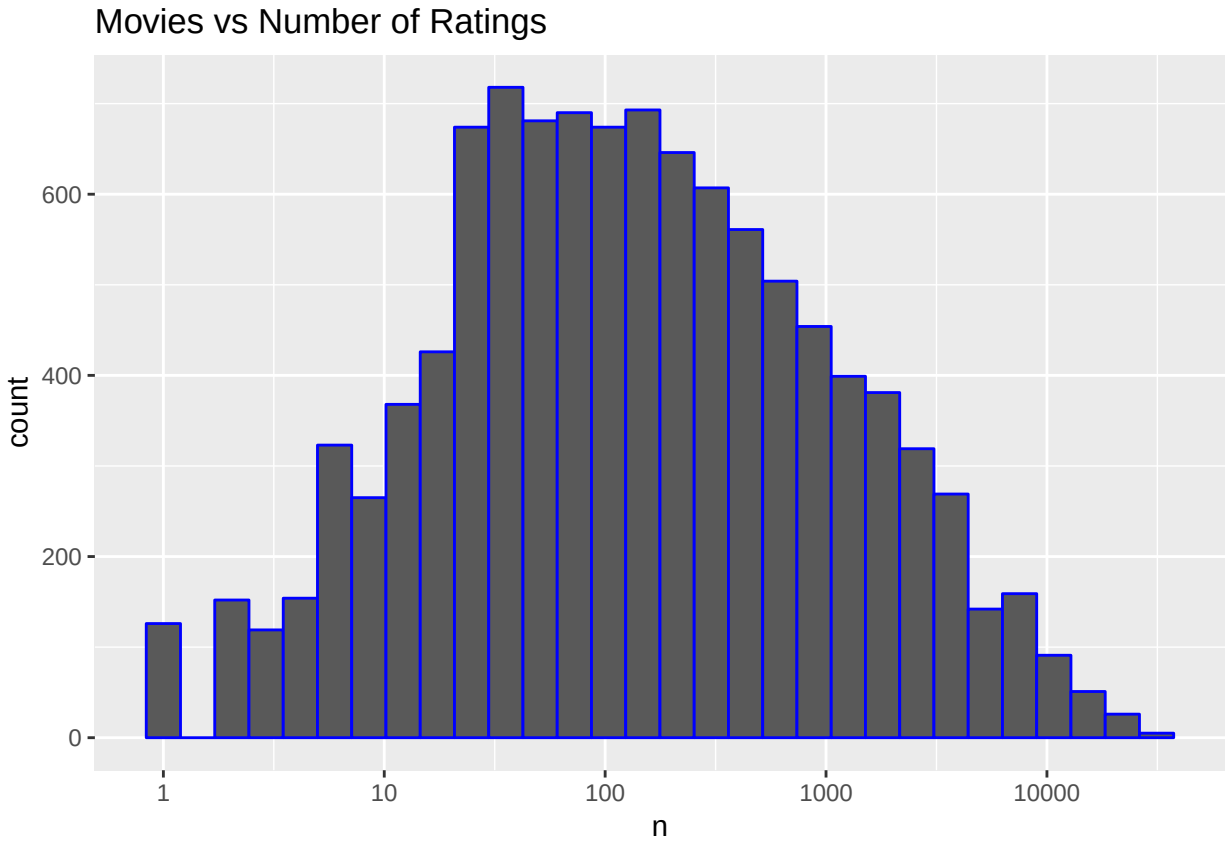
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



In general, recent users relatively rate movies lower.

Movies vs Number of Ratings

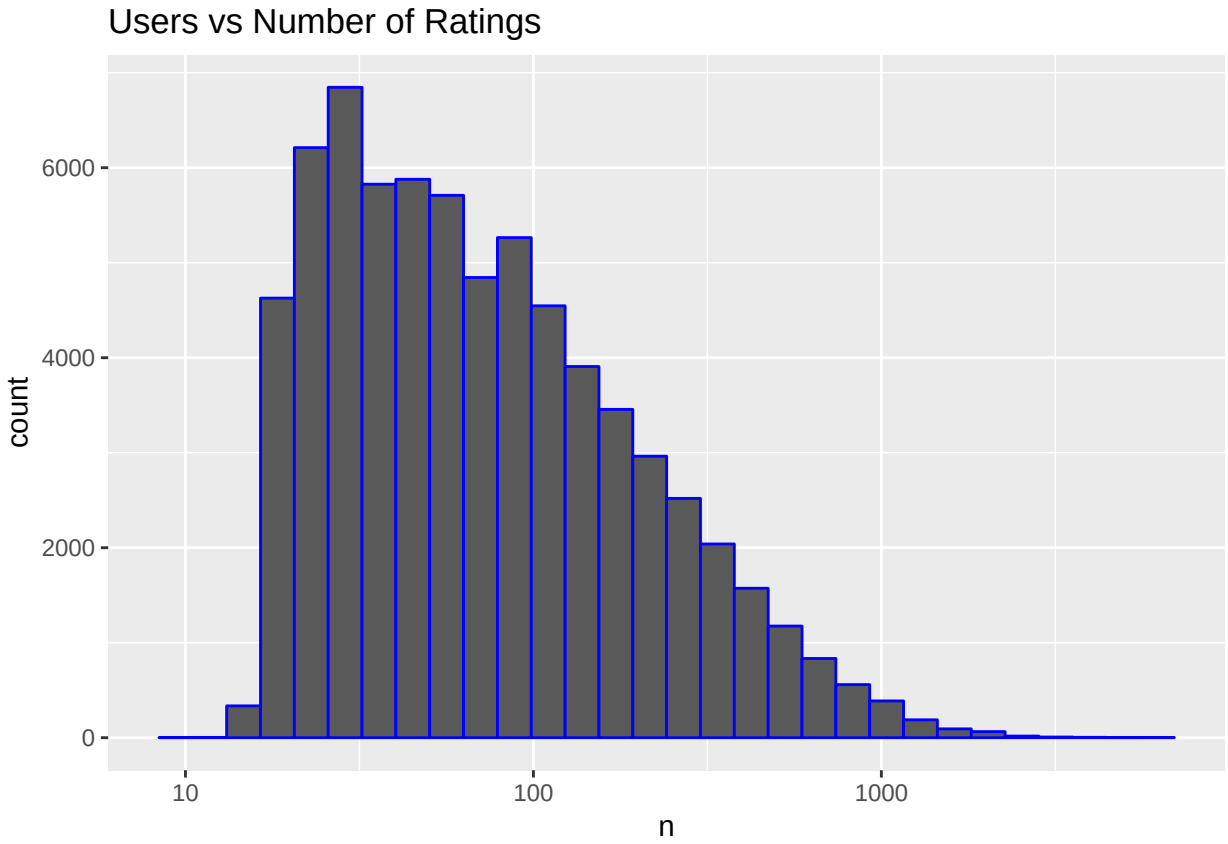
Movie biases can be exploited below (i.e. some movies get rated more than others):



The figure above shows that some blockbuster or great commercialized movies are viewed by millions of people while other low profile movies may be watched by few people.

Users vs Number of Ratings

User biases can be exploited below (i.e. some users rate movies more than others):

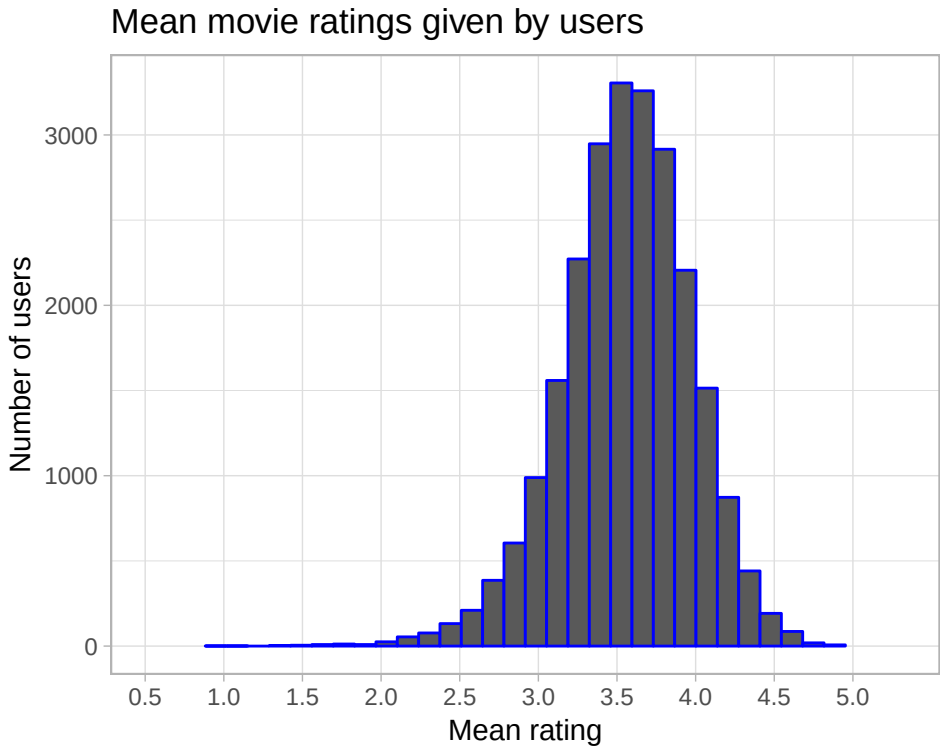


The figure above shows that some users are more active than others in rating movies. On average the number of users is inversely related with the number of ratings.

Average Ratings Distribution

The average rating for users who have rated at least 100 movies can be visualized below:

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "blue") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



The figure above shows that users mostly differ on how they are with their ratings.

Data Analysis

Loss Function

The RMSE metric indicates how good an algorithm performs in terms of predictions on a test set (or validation set). If $y_{u,i}$ is defined as the observed rating for movie i by user u and $\hat{y}_{u,i}$ is denoted as the predicted rating for movie i by user u. The RMSE is then defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

with N being the number of user/movie combinations and the sum occurring over all these combinations.

The RMSE can also be interpreted as a standard deviation: it is the typical error incurred when predicting a movie rating. If this number is larger than 1, it means the typical error is larger than one star, which is not good.The written function to compute the RMSE for vectors of ratings and their corresponding predictions is:

```
RMSE <- function(true_ratings, predicted_ratings){
sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Modelling Approach

I. Baseline model: Average movie rating

The baseline model is the simplest possible recommendation system: The same rating is predicted for all movies regardless of user. A model-based approach can be used to determine the specific number to be predicted. A model that assumes the same rating for all movies and users with all the differences explained by random variation is shown below:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

with $\epsilon_{u,i}$ independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies. This model makes the assumption that all differences in movie ratings are explained by random variation alone. It is known that the estimate that minimize the RMSE is the least square estimate of $Y_{u,i}$, in this case, is the average of all ratings:

```
mu_hat <- mean(edx$rating)
mu_hat
```

```
## [1] 3.512465
```

If all unknow ratings are predicted with mu_hat, the first naive RMSE is given by:

```
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

```
## [1] 1.061202
```

The results of the first RMSE is represented below:

```
rmse_results <- tibble(method = "Average movie rating model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.061202

This baseline RMSE will be compared with subsequent modelling approaches can be improved by accounting for features such as movie and user effects, and regularization.

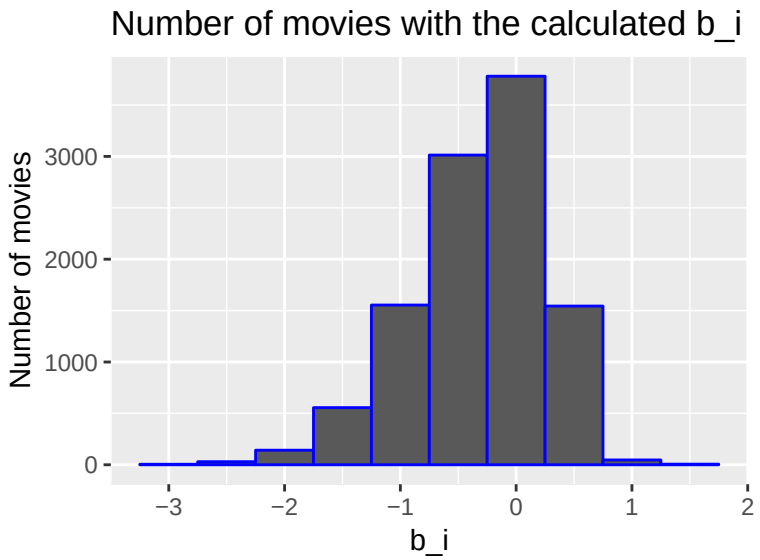
II. Movie Effect model

The baseline model can be improved by accounting for movie effects. It is known from experience that some movies are just generally rated higher than others. In general, the popular the movie the higher the movie. The baseline model can be augmented by adding the term bi to represent average ranking for movie i:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

Where bi is referred to as the "effects" or the bias term. A plot of the number of movies with the calculated bias, b_i is shown below:

```
mu <- mean(edx$rating)
avg_movie <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
avg_movie %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("blue"), ylab = "Number of mo
```



The above histogram is skewed to the left indicating that more movies have negative effects. This is referred to as the penalty term movie effect. The prediction below has a lower RMSE and, hence an improvement over the baseline model. However, the individual user rating is not yet accounted for.

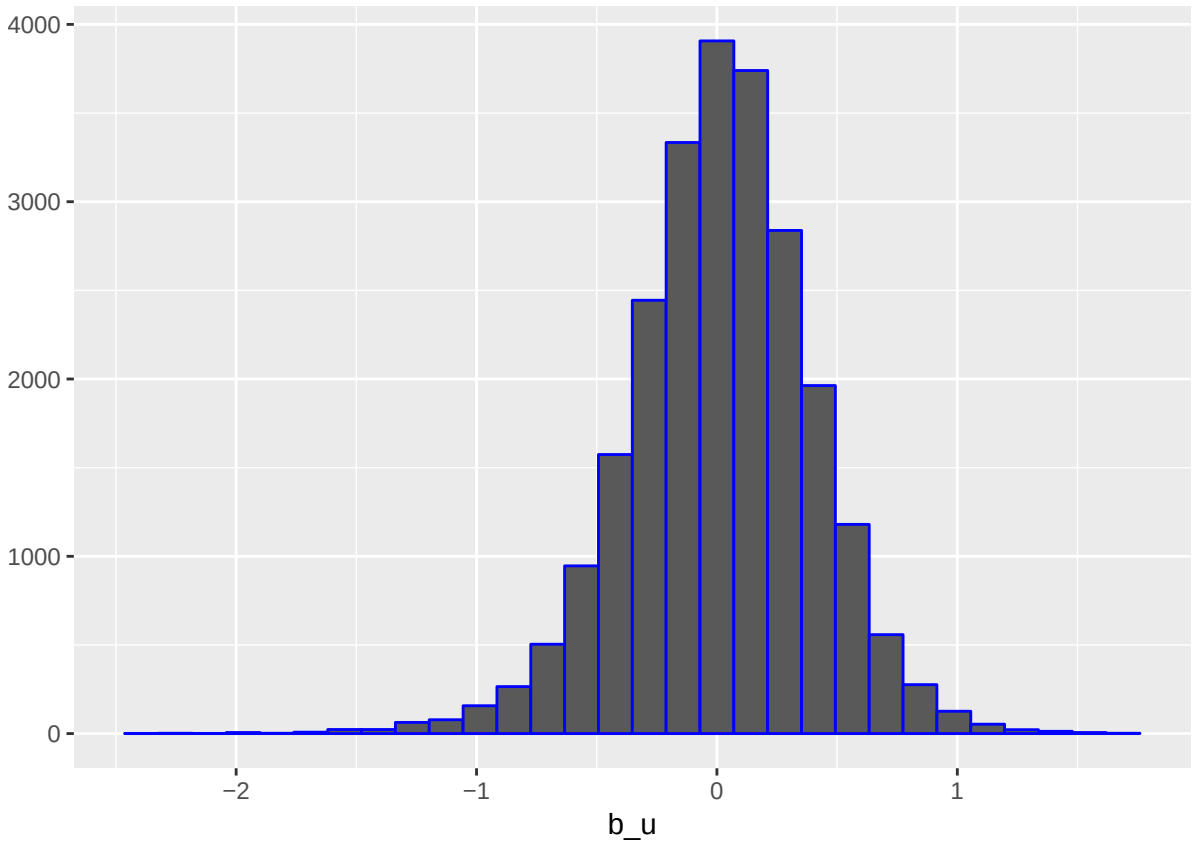
```
predicted_ratings <- mu + validation %>%
  left_join(avg_movie, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie effect model",
                                 RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

III. Movie and User Effect model

The movie effect model can be improved by accounting for user effects. The average rating for user u for those that have rated over 100 movies can be computed as follows:

```
avg_user <- edx %>%
  left_join(avg_movie, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
avg_user %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("blue"))
```



The above histogram shows that users can affect the ratings either positively or negatively. The histogram also shows that there is substantial variability across users: some users are very weird and others like every movie. This indicates that a further improvement to our model may be:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where b_u is a user-specific effect. If a weird user (negative b_u rates a great movie (positive b_i), the effects counter each other such that it can be correctly predicted that user gave that movie a 3 rather than a 5.

An approximation can be calculated by computing μ and b_i, and estimating b_u, as the average of

$$Y_{u,i} - \mu - b_i$$

```
avg_user <- edx %>%
  left_join(avg_movie, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

Predictors can now be constructed to see how much the RMSE improves:

```
predicted_ratings <- validation%>%
  left_join(avg_movie, by='movieId') %>%
  left_join(avg_user, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
```

```
                          tibble(method="Movie and User effect model",
                                 RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and User effect model	0.8653488

The result shows a good improvement over the movie effect model.

IV. Regularized Movie and User Effect model

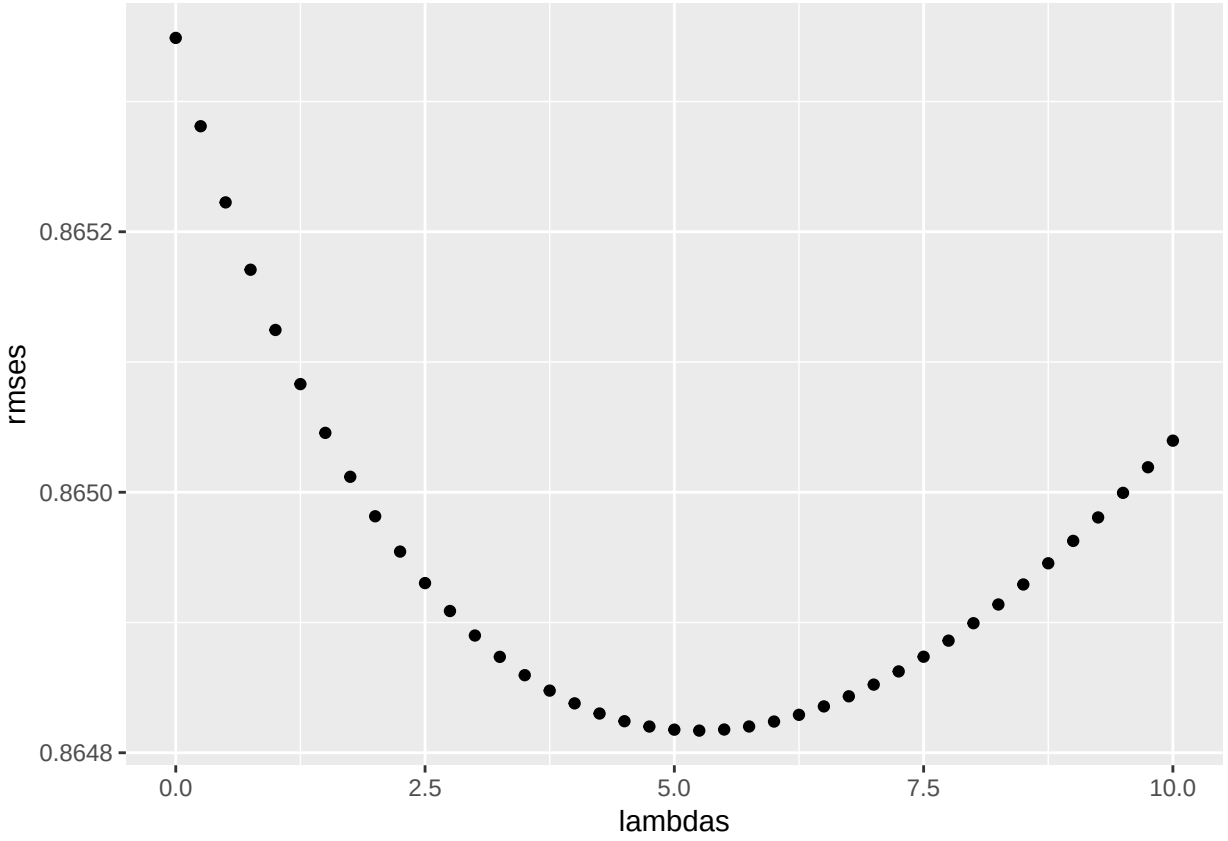
To resolve some of the mistakes in the previous models, the concept of regularization, that permits to penalize large estimates that come from small sample sizes is introduced. The main concept is to add a penalty for large values of bi to the sum of squares equation. The computed estimates of b_i and b_u are caused by movies with very few ratings and some users that only rated a very small number of movies, respectively. These estimates can strongly influence the prediction. The use of regularization permits to penalize these effects. Regularization method can also be used to reduce the effect of overfitting.

A tuning parameter called lambda, which will shrink the b_i and b_u in case of small number of ratings can be computed to minimize the RMSE.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
mu <- mean(edx$rating)
b_i <- edx %>%
group_by(movieId) %>%
summarize(b_i = sum(rating - mu)/(n()+l))
b_u <- edx %>%
left_join(b_i, by="movieId") %>%
group_by(userId) %>%
summarize(b_u = sum(rating - b_i - mu)/(n()+l))
predicted_ratings <-
validation %>%
left_join(b_i, by = "movieId") %>%
left_join(b_u, by = "userId") %>%
mutate(pred = mu + b_i + b_u) %>%
pull(pred)
return(RMSE(predicted_ratings, validation$rating))
})
```

A plot of RMSE vs lambdas is made to select the optimal lambda.

```
qplot(lambdas, rmses)
```



The optimal lambda that minimises the RMSE is:

```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

The final model result is shown below with the lowest RMSE.

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie and User effect model", RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and User effect model	0.8653488
Regularized Movie and User effect model	0.8648170

Results

RMSE Results

The RMSE values for the used models are shown below:

method	RMSE
Average movie rating model	1.0612018

method	RMSE
Movie effect model	0.9439087
Movie and User effect model	0.8653488
Regularized Movie and User effect model	0.8648170

The lowest value of RMSE that was computed is 0.8648170

Discussion

The final optimal model for the movieLens rating prediction project is the following:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

The results indicate that each model is an improvement from the previous one with the last one having an RMSE lower than 0.8649. There is a significant improvement from the baseline naive model which had a RMSE greater than 1. The regularized model made it possible for the machine learning algorithm to predict movie ratings for the movies in validation set.

Conclusion

A machine learning algorithm has been developed to predict movie ratings with MovieLens dataset. The baseline model which predicts the same rating (average rating) for all movies regardless of user computed a RMSE greater 1. RMSE greater than 1 means that a typical error is larger than one star, which is not good. By accounting for the movie effect, the RMSE went down to 0.9439087 which was an improvement over the baseline. After accounting for both movie and user effects, the RMSE further reduced to 0.8653488. However, the optimal model characterized by the lowest RMSE (0.8648170) was achieved by regularizing the movie and user effects, which penalized larger estimates of ratings from relatively small sample size of users. The initial goal of the project was eventually achieved by having a RMSE lower than 0.8649.

The optimal regularized model could further be improved by adding other effect such as genre, year, age and etc. but those models have to be computed on better hardware with better speed, good RAM and potential GPUs.

References

1. https://grouplens.org/datasets/movielens/10m/
2. http://files.grouplens.org/datasets/movielens/ml-10m.zip
3. https://rafalab.github.io/dsbook/